openmaru

JBoss EAP 7 Datasource

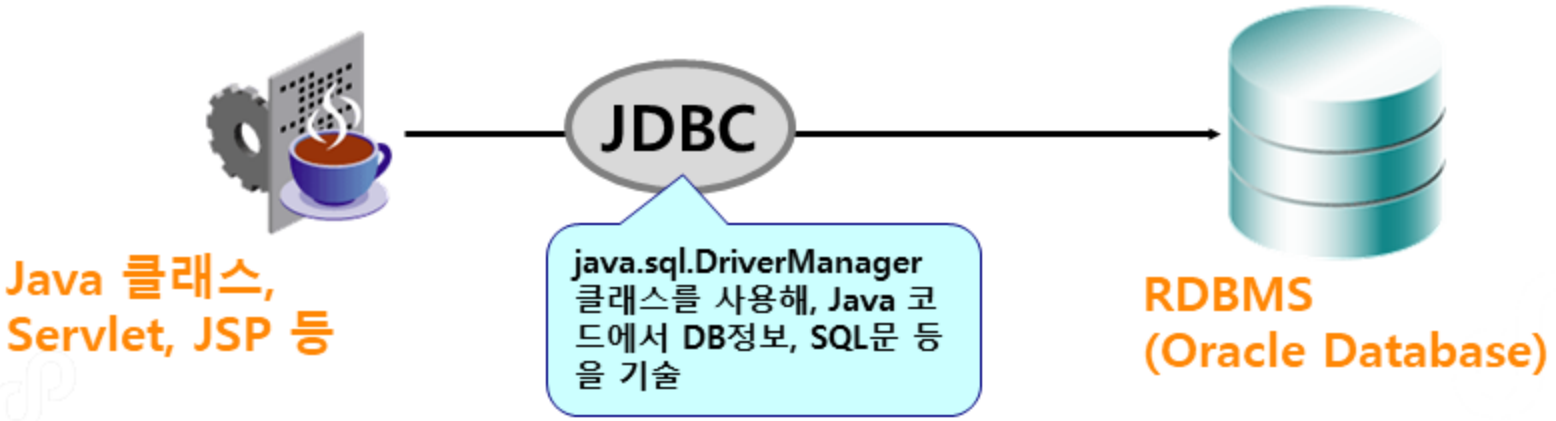
Opennaru, Inc.

Agenda

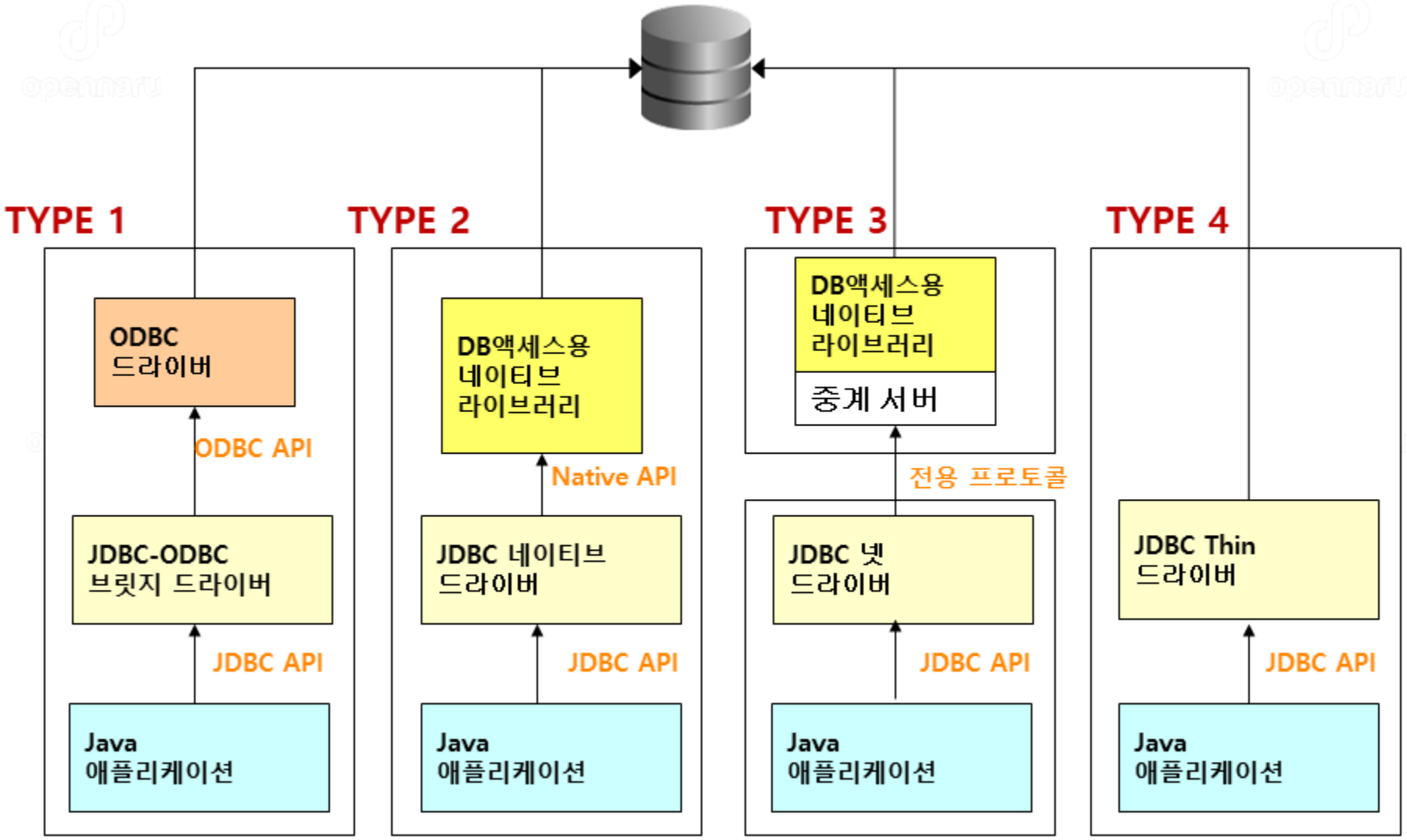
- 기본 개념
  - JDBC란?
    - JDBC Driver의 Type
    - SQL문 실행 절차
    - PreparedStatement Cache
    - JDBC DataSource란?
    - JDBC Connection Pool이란?
    - JDBC Connection Pool 설정 가이드
- JBoss EAP 7의 데이터소스
  - JBoss EAP 7의 Driver 배포방법
  - JBoss EAP 7 데이터소스 설정방법(웹 콘솔, CLI)
  - JBoss EAP 7 데이터소스 설정 옵션
  - Timeout 관련 파라미터
  - 장애감지 관련 파라미터
  - 설정 권고안
- 전자 정부 프레임워크 데이터 소스 설정방법

JDBC란

- JDBC는 Java Database Connectivity의 약자
- JDBC는, Java로부터 RDBMS에 접속하기 위한 표준 인터페이스
- DB 종류에 상관없이, 공통의 API를 이용해 데이터베이스를 조작하는 애플리케이션을 개발할 수 있다.
- JDBC의 API 패키지는 java.sql 및 javax.sql 패키지에 포함되어 있다.
- JDBC의 API를 사용하기 위해서는 JDBC Driver가 필요합니다. 이것은 실제로 데이터베이스 접속에 필요한 라이브러리로 데이터베이스 벤더가 제공한다.



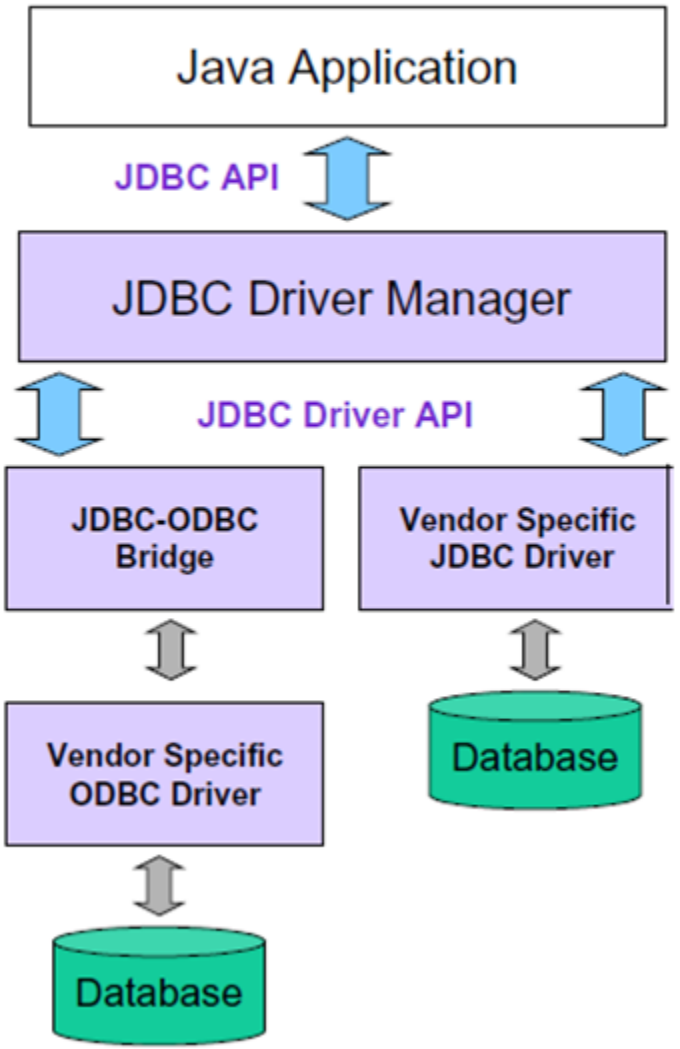
JDBC 드라이버의 타입



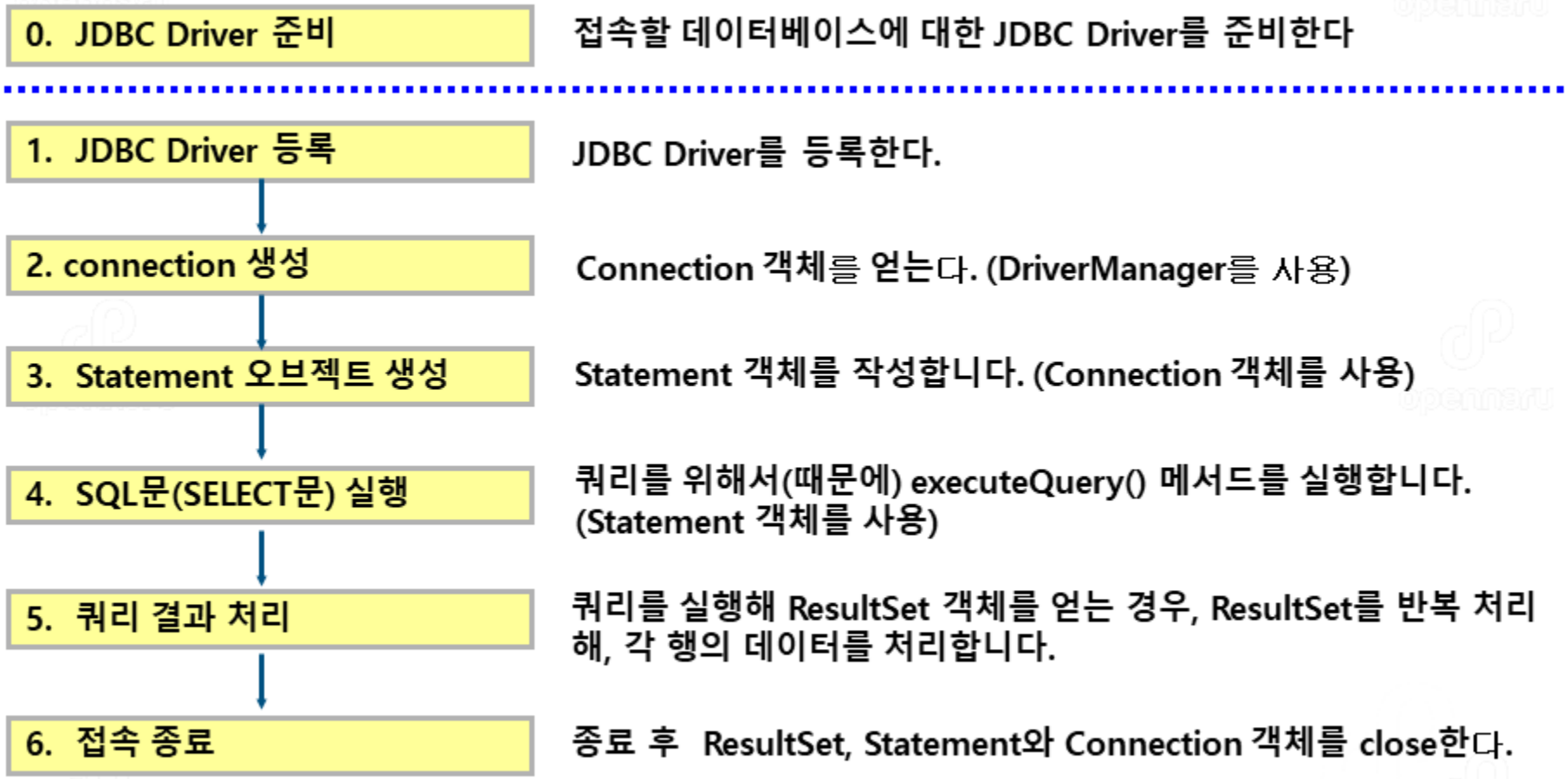
JDBC란?

JDBC가 하는일

- ☐ 데이터베이스 연결
- ☐ SQL문장 전송
- ☐ 결과 처리



JDBC를 사용한 SELECT문 실행



JDBC 코드 – Select

```
import java.sql. *;
...
String sql = "select * from emp";  //실행하는SQL문
DriverManager.registerDriver(new  oracle.jdbc.driver.OracleDriver());
String url = "jdbc:oracle:thin:hostname:1521:SID";
Connection conn =DriverManager.getConnection(url, "SCOTT" , "TIGER");  //connection
  취득
Statement stmt = conn.createStatement();   //Statement의 작성
ResultSet rset = stmt.executeQuery(sql);   //결과 세트의 취득
...
rset.close();   // ResultSet close 처리
stmt.close();   //Statement close 처리
conn.close();   //connection close 처리
```

데이터 소스를 사용한 DB액세스에 필요한 패키지 임포트

JDBC URL
접속할 데이터베이스 지정

1. JDBC Driver 등록
2. Connection 생성
3. Statement 생성
4. SQL문(SELECT문) 실행
5. 쿼리 결과 처리
6. 접속 종료

※ JDBC의 사용중에 에러가 발생했을 경우, 데이터베이스에 액세스하는 모든 메소드는 SQLException를 Throw 합니다.

PreparedStatement 캐시

- JDBC로, 조건치만 변화하는 같은 SQL를 반복해 실행하는 경우는, 일반적으로 PreparedStatement를 사용한다. 그 경우, DB측의 해석 처리 횟수가 줄어들기 때문에, 반복 실행하는 경우는 Statement 사용시부터 성능 향상을 기대할 수 있다.
- JBoss의 JDBC 데이터 소스에서는, 이 PrepareStatement나 CallableStatement를 캐시하는 기능을 제공한다.
- PrepareStatement를 사용하는 애플리케이션에서는 성능 향상을 기대할 수 있다.

Statement 사용시

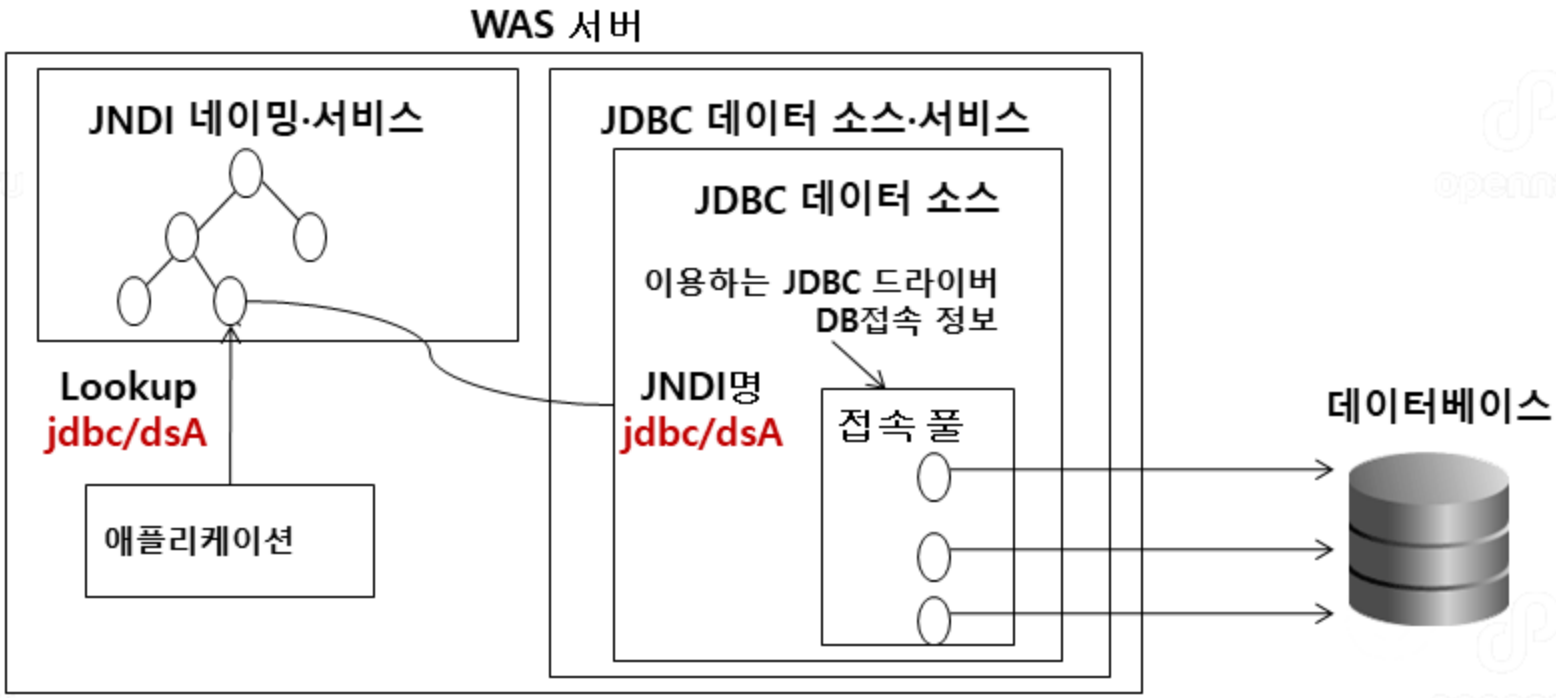
```
Statement stmt = conn.createStatement();
for ( int id = 0 ; id < 10000 ; id++ ) {
    String sql = "SELECT ENAME FROM EMP WHERE EMPNO = " + id;
    ResultSet rs = stmt.executeQuery(sql);
    while ( rs.next() ) {
      // 표시등의 처리
    }
}
```

PreparedStatement 사용시

```
String sql = "SELECT ENAME FROM EMP WHERE EMPNO = ?";
PreparedStatement    ps = conn.prepareStatement(sql);
for ( int id = 0 ; id < 10000 ; id++ ) {
    ps.setInt(1, id);
    ResultSet rs = ps.executeQuery();
    while ( rs.next() ) {
      // 표시등의 처리
    }
}
```

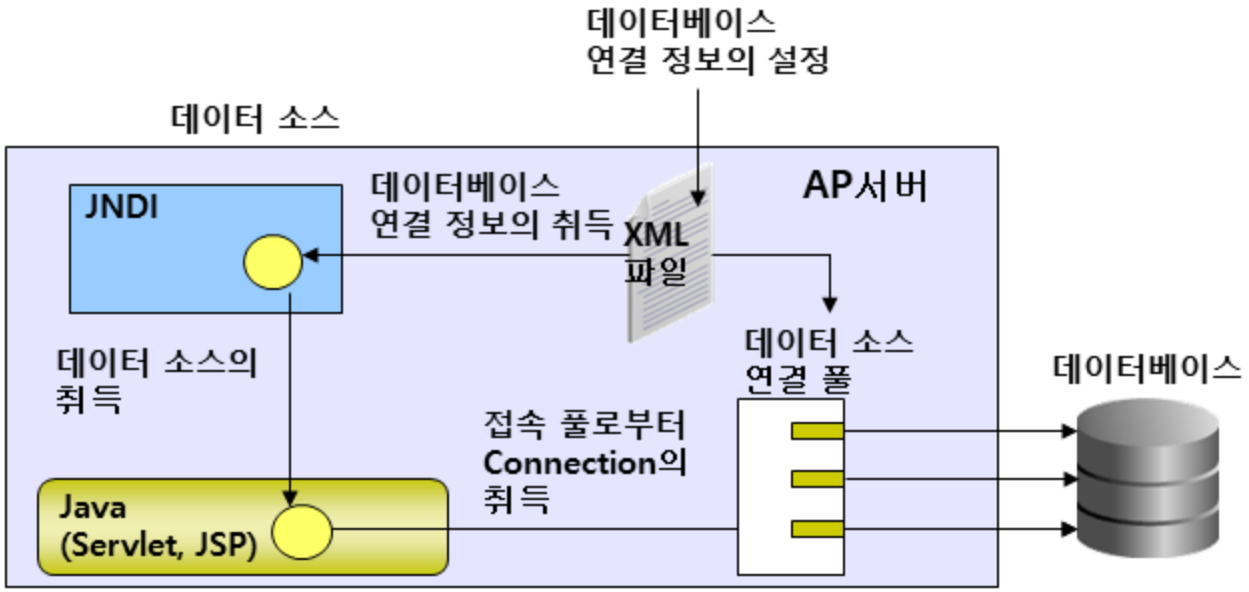
JDBC 데이터 소스란

- 애플리케이션 실행 환경(WAS)에서 애플리케이션에 데이터베이스 접속 서비스를 제공하는 기능
- 애플리케이션은, DB접속에 필요한 물리적인 정보(DB호스트명, DB유저 ID나 패스워드등)를 의식하지 않고 데이터베이스에 접속하여 쿼리를 실행할 수 있다.
- 접속 풀을 활용하는 것으로 DB접속, 종료 처리의 오버헤드를 없애 성능 향상 가능



데이터 소스란

- 일반적인 Java의 WAS서버에서, 데이터베이스와 미리 연결해 놓은 Connection 오브젝트를 여러 개 준비해 두어, 애플리케이션으로부터의 접속 요청시에 풀 안의 미리 연결된 Connection 오브젝트를 건네주는 접속 풀의 기능을 데이터 소스(JNDI)와 연결하여 제공한다.
- 데이터베이스 연결/종료 시간 성능 향상



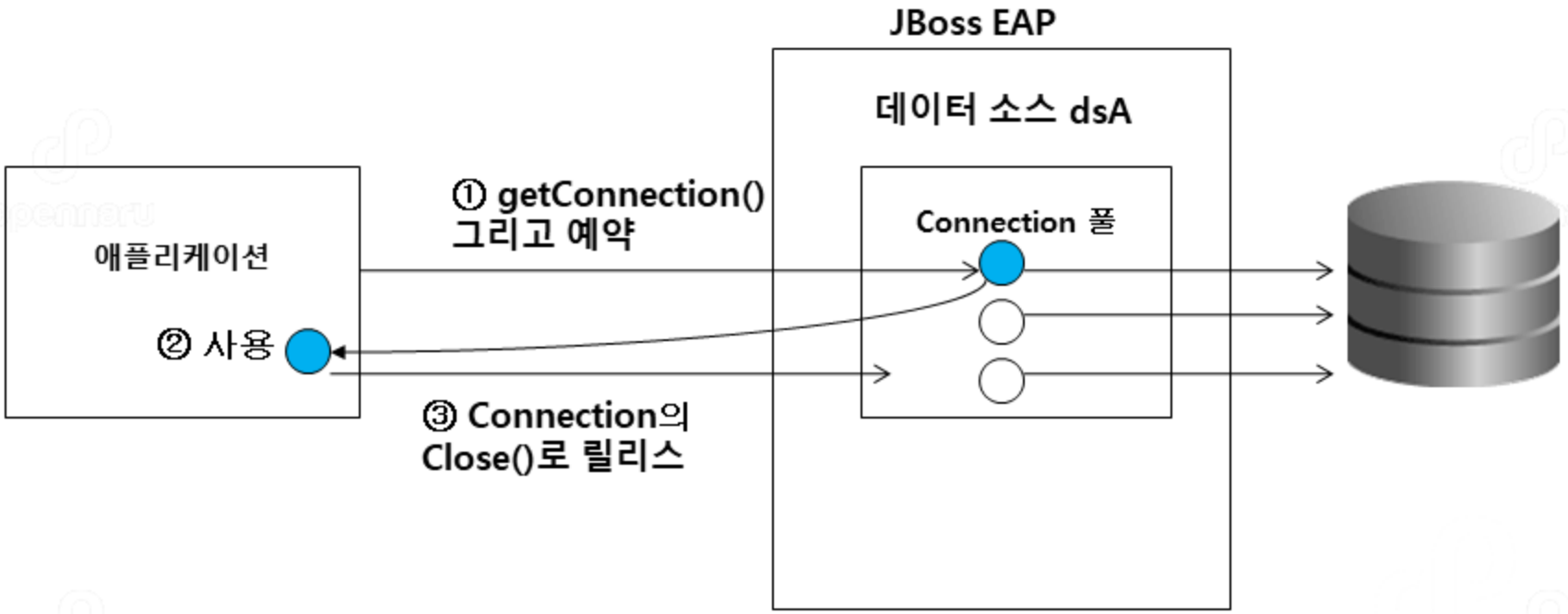
Connection Pool 이란

아래 코드는 JDBC 데이터 소스를 이용해 Connection 객체를 얻어 SQL문을 처리하는 예제

```
import javax.naming. *;
import javax.sql. *;
import java.sql. *;

...(중략)
String sql = "select * from emp"                          //실행할 SQL문
Context ic = new InitialContext();                        //JNDI 룩 업을 위한 Context
DataSource ds = (DataSource) ic.lookup("jdbc/dsA");       //데이터 소스 객체 얻기
Connection conn = ds.getConnection();                     //connection 객체 얻기
Statement stmt = conn.createStatement();                  //Statement의 작성
ResultSet rset = stmt.executeQuery(sql);                  // ResultSet 얻기

... ..(중략)

rset.close();  //resultset  close
stmt.close();  //Statement close
conn.close();  //connection close
```

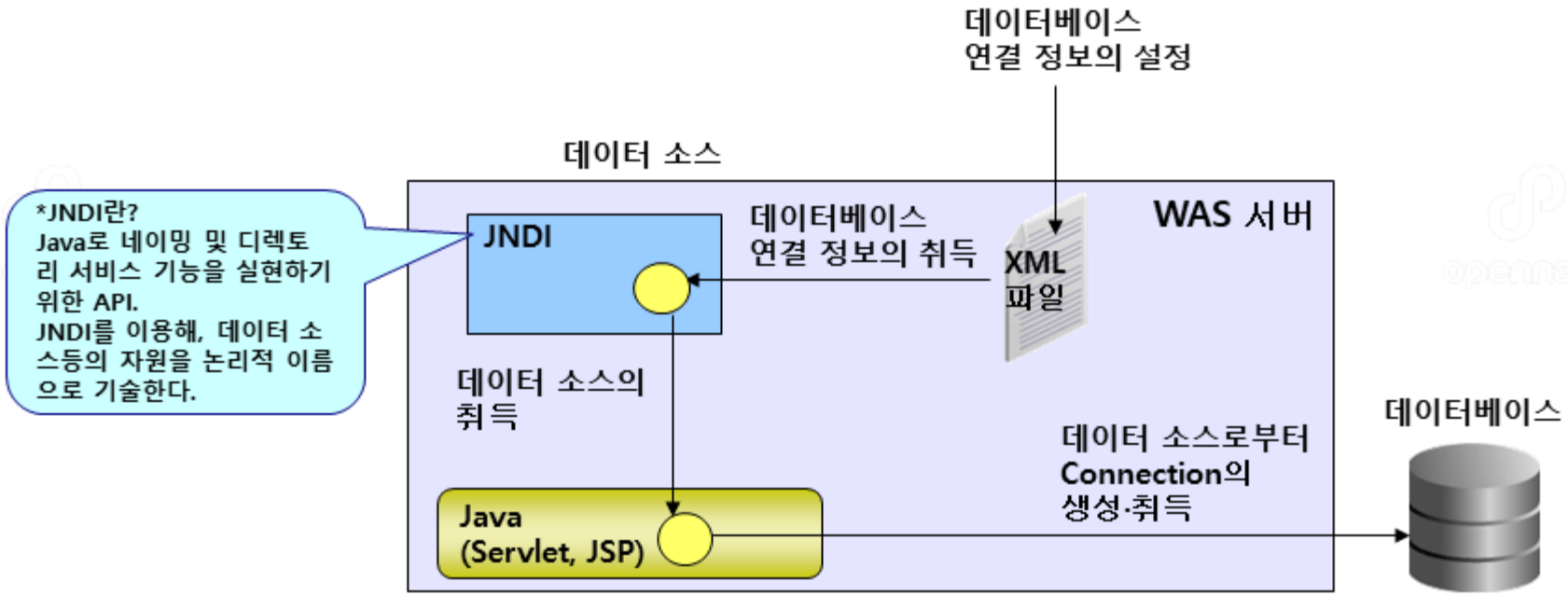
JDBC 데이터 소스를 이용하는 경우의 코드예

- 애플리케이션이 데이터소스를 이용해 Connection을 요청하면, 데이터 소스의 접속 풀에 사용하지 않는 Connection이 있으면, 그것을 애플리케이션에서 가져와 사용한다.
- 애플리케이션은 사용후 Connection을 Release(Close)하면, Connection 풀에 다시 넣는다.



Connection Pool 사용 방법

- javax.sql.DataSource 인터페이스를 구현한 Java 오브젝트(JDBC2.0부터 지원)
- 데이터 소스를 이용하면, JDBC 드라이버명, 접속 DB정보등의 환경에 의존하는 정보를 Java 코드에서 기술하지 않고, WAS 설정 파일에서 관리한다.



JDBC Connection 풀 설정 시 주의점

- **Connection 풀의 Min과 Max를 같게 한다.**
  - 접속 풀의 초기 용량과 최대 용량이 다른 경우, 동시 요청수가 많아지면 접속 풀 내에서 새롭게 DB에 접속이 발생한다. 또, 사용하지 않을 경우 접속 풀내의 연결을 제거한다. 이 연결 처리가 빠르지 않다.
  - 그 때문에 운영 환경에서는 Connection 풀 안의 Connection이 증감하지 않게 「 초기 갯수」와 「최대 개수」를 같게 설정하는 것을 추천한다.
- **JBoss에서 동시 실행되는 최대 thread수 <= 커넥션 풀 「최대값」**
  - JBoss의 애플리케이션은 thread에서 처리되지만, 동시 실행 thread수가 많아지는 경우 커넥션 풀에 가용한 커넥션이 없으면 thread가 대기하게 되어 성능에 큰 영향을 미친다.
  - 최대 thread수=특정의 접속 풀의 최대 용량으로 지정한다.
- **커넥션에 대한 테스트**
  - 데이터베이스나 서버측의 부하(CPU 사용율등)가 매우 높은 경우, 접속 풀의 테스트 기능을 사용하지 않아도 된다.
  - 데이터베이스 장애나 네트워크 장애가 많은 환경이라면 반드시 사용한다.
- **WAS와 DB 간 방화벽 설정**
  - 방화벽에 따라서는, Idle 상태인 통신을 자동으로 끊는 기능을 제공하지만, 접속 풀은 DB와의 통신을 계속 유지하기 때문에 영향을 받을 가능성이 있다.

JDBC Driver의 선택

Oracle JDBC Driver의 선택 가이드

- **Oracle의 경우 RAC TAF를 사용하지 않는다면 Thin 드라이버를 사용하라**
- **Oracle JDBC Driver 호환성 테이블**
  - http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-faq-090281.html#02_02
  - 드라이버 버전 별 사용 가능한 DB
    - 11 Driver → 11g, 10g, 9i DB
    - 9.2 ~ 10.2 Driver → 11g, 10g, 9i, 8 DB
    - 10.1 Driver → 11g, 10g, 9i, 8, 7 DB
  - JDK 버전별 Driver 파일명
    - ojdbc5.jar → JDK 1.5용, ojdbc6.jar → JDK 1.6용, classes111.jar → JDK 1.1용
    - classes12.jar → JDK 1.2, 1.3용, ojdbc14.jar → JDK 1.4용
    - *_g.jar 파일 → 디버그정보를 포함한 드라이버
- **OCI 드라이버 → libocijdbc<major_version_id>.so 버전번호가 표시됨**
- **데이터베이스 벤더에서 제공하는 최신의 JDBC 드라이버를 사용**

JBoss EAP 7 DataSource : DataSource 개요

- **DataSource 란**
  - **JDBC Spce ver 2.0 에 대해 추가된 JDBC 기능**
    - ✓ DataSource 로부터 Connection 를 가져오는 방식
    - ✓ Connection 풀링 기능을 이용
    - ✓ DataSource 는 JNDI 로 lookup 를 하여 가져옴

- **JBoss EAP6에서 DataSource 설정은 2 단계로 진행**
  - **JDBC 드라이버 설치**
    - ✓ 애플리케이션과 같이 JDBC 드라이버를 배포
    - ✓ **Module 방식으로 설치**
  - **데이터 소스의 설정**
    - ✓ 웹 콘솔에서 설정
    - ✓ CLI 에서 설정
    - ※ JBoss EAP 5. x 와 같이 ds.xml 에 의한 설정도 가능하지만 추천하지 않음

JBoss EAP 7 DataSource: JDBC 드라이버 배포 방법

- **JBoss EAP 6는 모듈기반이기 때문에 아래의 2가지 방법 중 하나를 선택하여 배포**

1. JDBC 드라이버 라이브러리를 배포

- **Deployment에서 언급한 내용과 동일**
  - ✓ Deployment Scanner에서 배포
  - ✓ 웹 콘솔에서 배포
  - ✓ CLI (Command Line Interface) 에서 배포

    예) jboss-cli.sh -c --command="deploy ../../mysql-connector-java-5. 1.6. jar"

    ※ 이 방식은 JDBC 4-compliant JDBC 드라이버만 가능

2. Module 방식 배포

- **$JBOSS_HOME/modules/xxx/main에 . jar 와 module.xml 를 저장**

JBoss EAP 7 DataSource: JDBC 드라이버를 모듈 방식으로 배포

- . jar 를 모듈 형태로 작성하여 $JBOSS_HOME/modules/system/layers/ext 디렉터리에 복사
- 모듈 작성 순서

1. module 명의 결정 (임의로 모듈명을 "com.mysql" 으로 사용)
2. jdbc 드라이버 복사 ($JBOSS_HOME/modules/system/layers/ext)
   - ✓ module 명이 "com.mysql"이므로 아래와 같이 드라이버를 배치
     - ✓ $JBOSS_HOME/modules/system/layers/ext/com/mysql/main/**mysql-connector-java-5.1.6.jar**
3. module.xml 작성하여 위의 디렉터리에 복사
   - ✓ $JBOSS_HOME/modules/system/layers/ext/com/mysql/main/module.xml
4. JDBC 드라이버가 javax.naming 를 사용하기 때문에 javax.api 에의 의존성을 선언
5. driver 선언
   - ✓ 서버 / 도메인 설정 파일에 내용 추가
   - ✓ $JBOSS_HOME/standalone/configuration/standalone*.xml
   - ✓ $JBOSS_HOME/domain/configuration/domain.xml

$JBOSS_HOME/modules/com/mysql/main/module.xml

```
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.1.6.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
  </dependencies>
</module>
```

standalone.xml

```
<subsystem xmlns="urn:jboss:domain:datasources:1.1">
 <datasources>
  <drivers>
    <driver name="com.mysql" module="com.mysql">
      <xa-datasource-class>
        com.mysql.jdbc.jdbc2.optional.MysqlXADataSource
      </xa-datasource-class>
    </driver>
  </drivers>
 <datasources>
</subsystem>
```

JBoss EAP 7 DataSource: DataSource 정의

- DataSource 정의는 Standalone/domain 설정 파일에서 작성
  - $JBOSS_HOME/standalone/configuration/standalone.xml
  - $JBOSS_HOME/domain/configuration/domain.xml
- DataSource 정의 예는 다음과 같은
  - <datasource> Element 정의

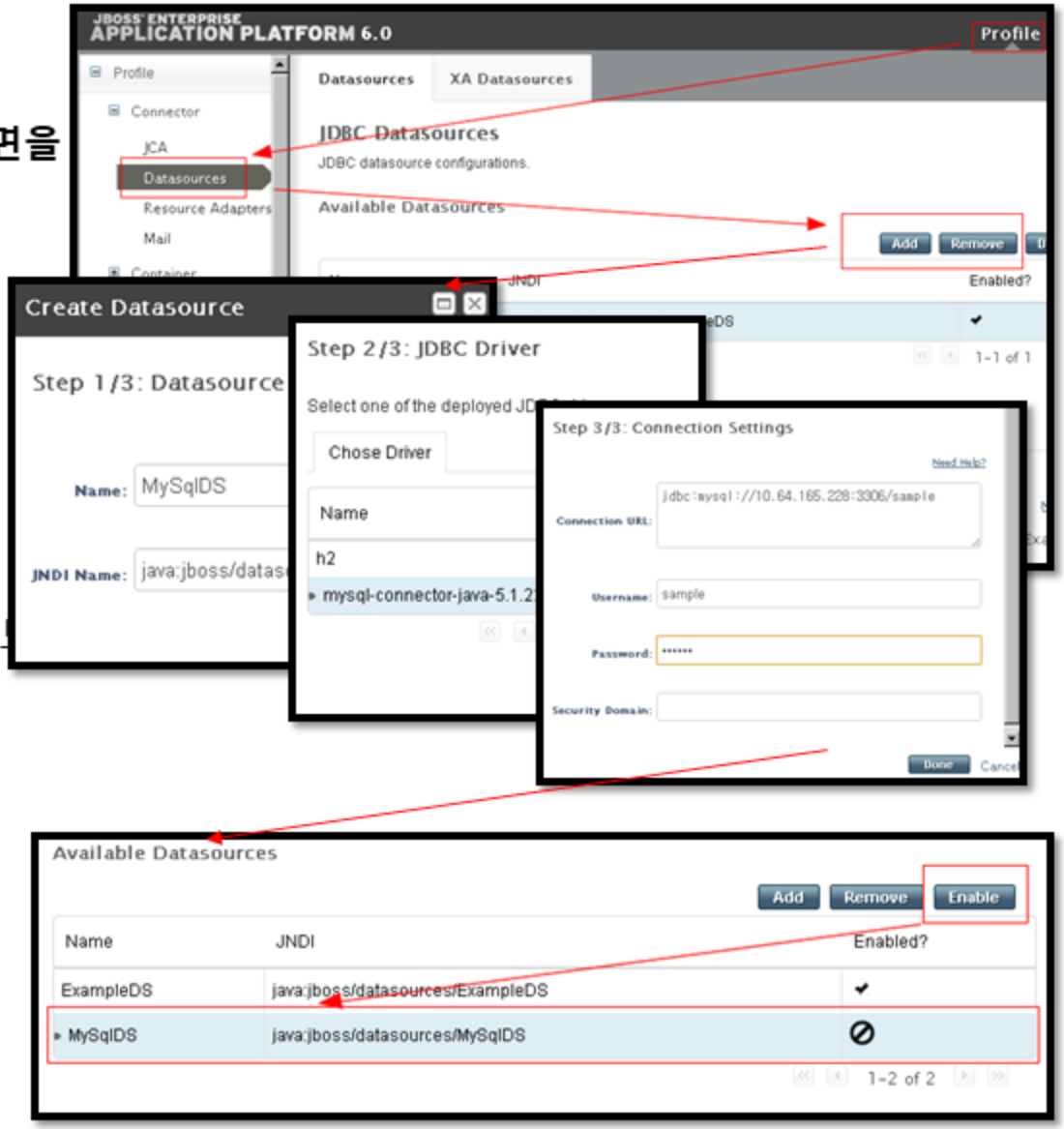
standalone.xml

```
<subsystem xmlns="urn:jboss:domain:datasources:1.0">
  <datasources>
     <datasource jndi-name="java:jboss/datasources/MySqlDS" pool-name="MySqlDS">
       <connection-url>jdbc:mysql://localhost:3306/sample</connection-url>
              <driver>com.mysql</driver>
              ...
              <security>
                <user-name>sample</user-name>
                <password>sample</password>
              </security>
              ...
    </datasource>
    <drivers>
              <driver name="com.mysql" module="com.mysql">
              ...
    </drivers>
  </datasources>
</subsystem>
```

driver 선언

JBoss EAP 7 DataSource: 웹 콘솔에서 데이터소스 설정

- 웹 콘솔 접속
  - 관리 콘솔에 로그인해 「Profile」로 「Connector」의 「Datasources」화면을 표시
- 데이터 소스 추가
  - 다음의 필수 항목 입력
    - ✓ 데이터 소스명
    - ✓ JNDI 명
    - ✓ JDBC 드라이버 선택
    - ✓ DB 커넥션 정보/사용자/패스워
- Enable
  - 사용하려면 「Enable」를 클릭



JBoss EAP 7 DataSource: Standalone에서 CLI로 데이터소스 설정

- Standalone 모드에서 데이터소스 설정 하기
- JDBC 드라이버 배포 및 제거
  - deploy /rhproducts/download/mysql-connector-java-5.1.22-bin.jar
    - deploy ../../samples/ojdbc6.jar –name=ojdbc6.jar
  - undeploy mysql-connector-java-5.1.22-bin.jar
- CLI에서 데이터소스 관련 명령어
  - data-source enable --name=MySqlDS
  - data-source disable --name=MySqlDS
  - data-source remove --name=MySqlDS

```
[standalone@0.0.0.0:9990 /] deploy /rhproducts/download/mysql-connector-java-5.1.22-bin.jar
[standalone@0.0.0.0:9990 /] data-source add --name=MySqlDS ₩
>  --jndi-name=java:/jboss/datasources/MySqlDS ₩
>  --connection-url=jdbc:mysql://localhost:3306/sample ₩
>  --driver-name=mysql-connector-java-5.1.22-bin.jar ₩
>  --user-name=sample --password=sample
[standalone@0.0.0.0:9990 /] data-source enable --name=MySqlDS
```


JBoss EAP 7 DataSource: 도메인에서 CLI로 데이터소스 설정

- 도메인 모드에서는 profile을 선택해서 필요한 데이터소스를 정의

Standalone 모드

```
data-source add --name=[데이터 소스명] ₩
--jndi-name=[JNDI명] ₩
--connection-url=[접속 URL] ₩
--driver-name=[드라이버명] ₩
--user-name=[접속 사용자명] ₩
--password=[접속 사용자 패스워드]
```

도메인 모드

```
data-source add --name=[데이터 소스명] ₩
--profile=[설정 대상 프로파일] ₩
--jndi-name=[JNDI명] ₩
--connection-url=[접속 URL] ₩
--driver-name=[드라이버명] ₩
--user-name=[접속 사용자명] ₩
--password=[접속 사용자 패스워드]
```

```
[standalone@0.0.0.0:9990 /] deploy /rhproducts/download/mysql-connector-java-5.1.22-bin.jar
[standalone@0.0.0.0:9990 /] data-source add --name=MySqlDS --profile=full ₩
>  --jndi-name=java:/jboss/datasources/MySqlDS ₩
>  --connection-url=jdbc:mysql://localhost:3306/sample ₩
>  --driver-name=mysql-connector-java-5.1.22-bin.jar ₩
>  --user-name=sample --password=sample
[standalone@0.0.0.0:9990 /] data-source enable --name=MySqlDS --profile=full
```

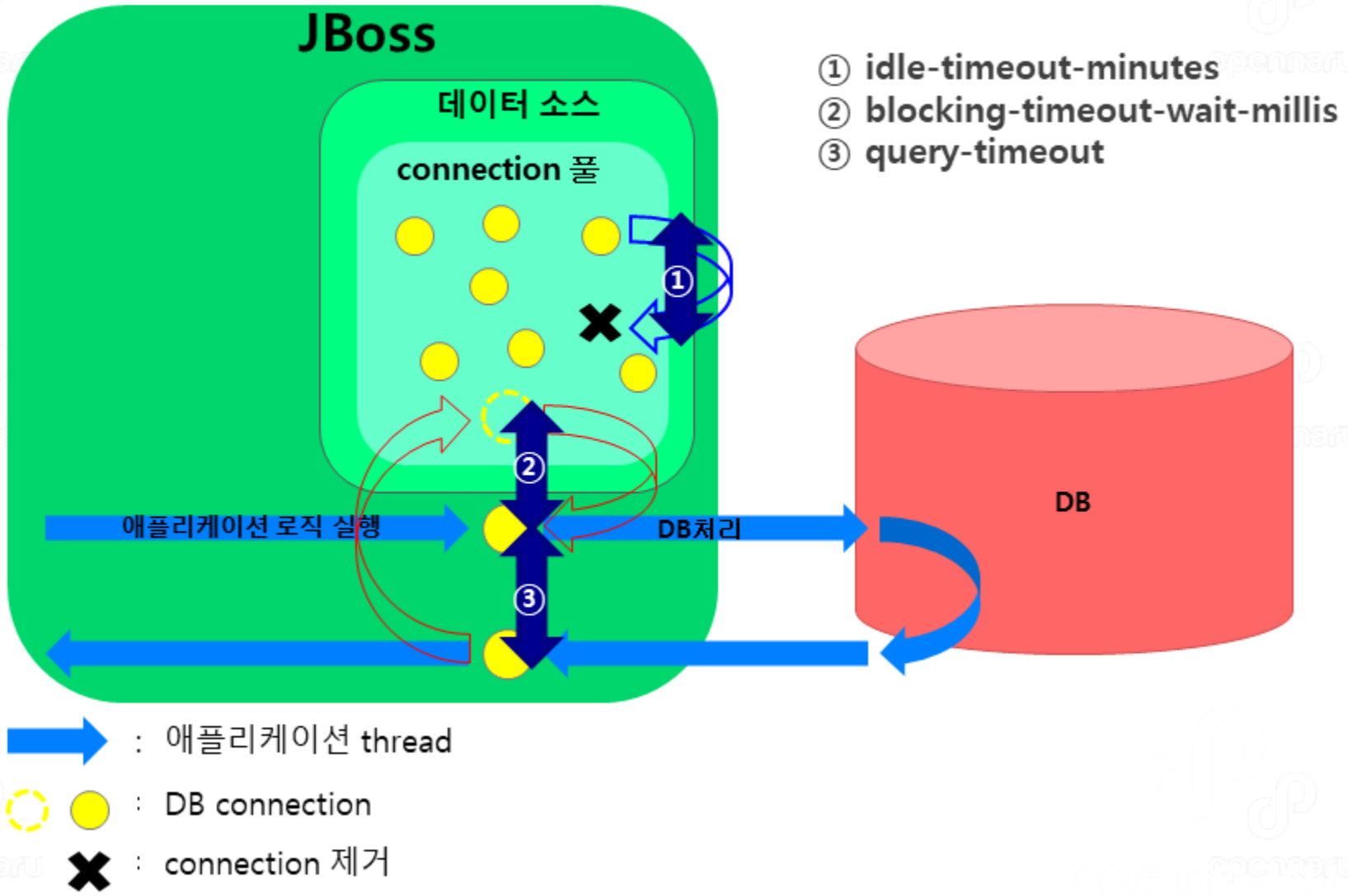
JDBC 데이터 소스 – Connection Pool 공통 파라미터

항목	설명	기본값
min-pool-size	풀을 유지하는 최소 연결 수	
max-pool-size	풀에서 유지 가능한 최대 연결 수	
prefill	연결 풀을 미리 채우도록 설정하는 설정. ● Prefill 이 true 인 경우에는 min-pool-size 값까지 풀의 커넥션이 연결된다.	false
use-strict-min	pool-size가 정확하게 생성되어 있는지 확인. 기본값은 false 로 설정되어 있다.	false
flush-strategy	오류가 있는 경우 풀을 갱신할지 여부를 설정한다. 설정가능한 옵션은 다음과 같다. ● FailingConnectionOnly : connection 에러가 발생한 connection만 삭제 ● IdleConnections : Idle 상태의 커넥션을 삭제 ● EntirePool : 모든 커넥션을 삭제	FailingConnectionOnly
allow-multiple-users	여러 사용자가 getConnection (user, password) 메소드를 사용 데이터 소스에 액세스하거나, 내부 풀 타입이 동작을 여부를 지정한다.	

JDBC 데이터 소스 – Timeout 관련 파라미터

항목	설명	기본값
blocking-timeout-millis	연결 대기 중에 블록하는 최대 시간 (밀리 초). 이 시간을 초과하면 예외가 발생된다. 기본값은 30000 (3 분)이다.	30000
idle-timeout-minutes	사용하지 않는 커넥션을 정기적으로 삭제한다. 기본값은 30이며 30 분 동안 이용하지 않는 커넥션이 삭제된다. 커넥션 이용여부 검사는 (지정된 값 / 2) 간격으로 실행되므로 15 분마다 확인하는 것이다.	30 (분)
set-tx-query-timeout	트랜잭션 시간 초과 될 때까지 남은 시간을 바탕으로 쿼리 제한 시간을 설정.	false
query-timeout	쿼리 제한 시간 (초). 기본값은 제한 없음이다.	0
allocation-retry	예외를 전달하기 전에 다시 커넥션을 시도하는 횟수. 기본값은 0 으로, 첫 실패 후 예외가 전달된다	0
allocation-retry-wait-millis	연결 할당까지 대기하는 시간 (밀리 초). 기본값은 5000에서 5 초이다.	5000
xa-resource-timeout	XAResource.setTransactionTimeout 메서드에 전달	

데이터소스 타임아웃 시간

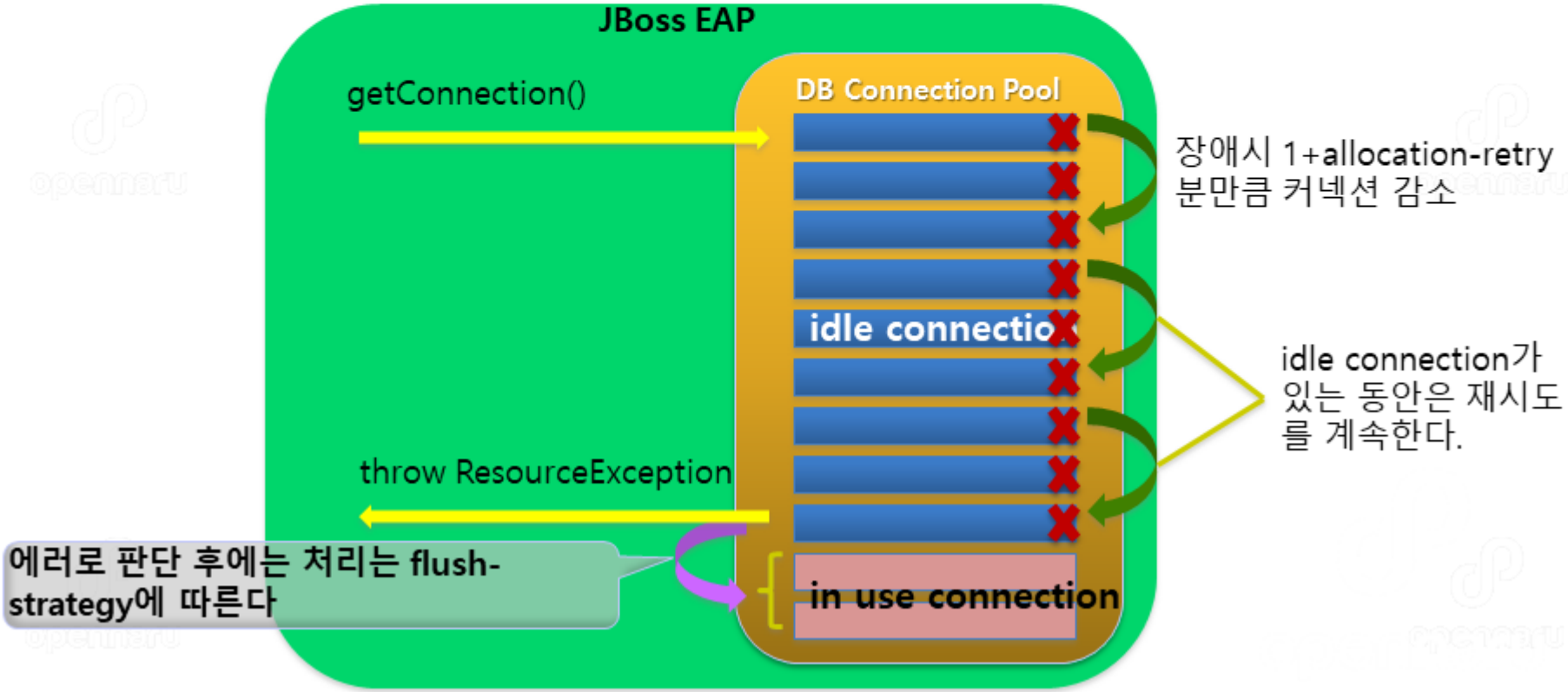


JDBC 데이터 소스 - 장애 감지를 위한 파라미터

항목	설명	기본값
valid-connection-checker	• SQLException.isValidConnection (Connection e) 설정을 전달 연결 검증 • org.jboss.jca.adaptors.jdbc.ValidConnectionChecker 인터페이스의 구현. • 연결이 끊기면 예외가 발생. • 이 파라미터는 check-valid-connection-sql 보다 이 파라미터가 우선	
check-valid-connection-sql	• 커넥션 풀의 유효성을 확인하는 SQL 문.	
validate-on-match	• 커넥션을 얻으려는 시점 (DataSource.getConnection() 때)에서 커넥션을 검사하는 옵션으로 true로 지정하는 것을 권장 • background validation 은 상호 배타적이다	False
background-validation	• 커넥션 연결 시점에서 확인 하는 것이 아니라 백그라운드 스레드로 커넥션을 확인 • validate-on-match 는 상호 배타적	
background-validation-millis	• 0보다 큰 값을 지정하여 백그라운드에서 커넥션을 검사(분 단위) 한다. 커넥션 풀에서 사용 중인 커넥션은 지정된 밀리 초 간격으로 정기적으로 연결을 체크한다. 기본값인 0은 동작을 하지 않도록 설정되어 있다.	0
use-fast-fail	• <validate-on-match> 옵션을 true로 하여 사용하면 풀에서 가져온 커넥션을 사용할 수 없는 경우 풀에서 다음 커넥션을 가져 오는 것이 아니라, 즉시 새로운 연결을 생성한다. 예를 들어 데이터베이스를 다시 시작하는 경우와 같이 모든 커넥션을 풀을 사용할 수 없게 되는 경우 빨리 연결을 할 수있도록 하는 옵션이다. 하지만 다른 커넥션에 대해 체크를 하지 않기 때문에 사용할 수 없는 커넥션은 풀에 남아 있게 된다. <use-fast-fail>을 비활성화하면 반대로 동작하게 되므로 모든 커넥션을 사용할 수 없는지 확인하고 삭제 한 후 새로운 연결을 하나 만들 수 있다. 이 경우 풀의 연결은 하나이며, <min-pool-size> 만들지 않는다. 하지만 백그라운드에서 커넥션이 삭제되는 시나리오에서는 <min-pool-size>까지 회복하게 된다.	false
flush-strategy	• 오류가 있는 경우 풀에 대한 Flush 전략	
stale-connection-checker	• org.jboss.jca.adapters.jdbc.StaleConnectionChecker 인스턴스에서 Boolean isStaleConnection (SQLException e) 메서드를 전달한다. 이 메소드가 true 를 돌려주는 경우, 예외는 SQLException 의 하위 클래스인 org.jboss.jca.adapters.jdbc.StaleConnectionException 로 포함되었다..	
exception-sorter	• org.jboss.jca.adapters.jdbc.ExceptionSorter 인스턴스에서 Boolean 변수 isExceptionFatal (SQLException e) 메서드를 전달한다. 이 메서드는 예외가 connectionErrorOccurred 메시지로 javax.resource.spi.ConnectionEventListener 의 모든 인스턴스에 브로드캐스트 할지를 확인한다.	

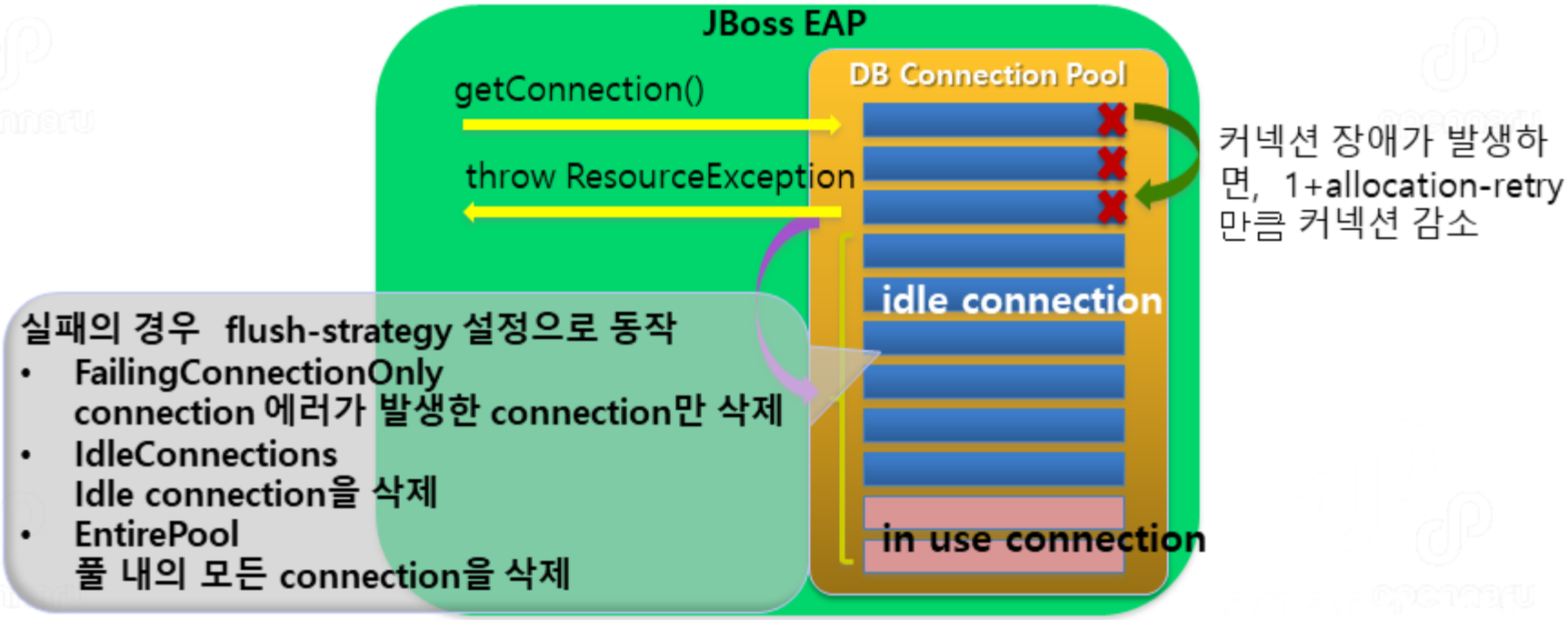
JDBC Connection Validation

- "use-fast-fail=false" 인 경우
  - Idle Connection가 존재하는 동안 계속 재시도
  - Idle Connection를 모두 시험해도 올바른 connection이 설정되지 않는 경우 flush-strategy로 결정



JDBC Connection Validation

- "use-fast-fail=true" 인 경우
  - <validate-on-match> 옵션을 true로 하여 사용하게 되면 풀에서 가져온 커넥션을 사용할 수 없는 경우 풀에서 다음 커넥션을 가져 오는 것이 아니라, 즉시 새로운 연결을 생성
  - 커넥션 생성에 실패 한 후에는 flush-strategy 설정으로 동작



Oracle 데이터소스 설정 권고안

```
<datasource jndi-name="java:jboss/datasources/testDS" pool-name="testDS" enabled="true" use-ccm="false">
    <connection-url>jdbc:oracle:thin:@localhost:1521:orcl</connection-url>
    <driver-class>oracle.jdbc.OracleDriver</driver-class>
    <driver>oracle</driver>
    <pool>
        <min-pool-size>150</min-pool-size>
        <max-pool-size>250</max-pool-size>
        <prefill>true</prefill>
        <use-strict-min>true</use-strict-min>
    </pool>
    <security>
        <user-name>userid</user-name>
        <password>passwd</password>
    </security>
    <validation>
        <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker"/>
        <stale-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectionChecker"></stale-connection-checker>
        <validate-on-match>false</validate-on-match>
        <background-validation>true</background-validation>
        <background-validation-millis>30000</background-validation-millis>
        <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter"/>
    </validation>
    <timeout>
        <blocking-timeout-millis>60000</blocking-timeout-millis>
        <idle-timeout-minutes>15</idle-timeout-minutes>
    </timeout>
    <statement>
        <track-statements>true</track-statements>
        <prepared-statement-cache-size>200</prepared-statement-cache-size>
        <share-prepared-statements>true</share-prepared-statements>
    </statement>
</datasource>
```

max는 Thread Pool의 개수만큼 설정한다.

prefill을 설정하면 Startup시 연결을 맺는다.
use-strict-min을 설정하여 최소값을 지킨다.

연결 상태 체크를 위해 SQL문을 사용하지 않고 Valid Checker를 사용하면 속도가 빨라진다.

지정된 시간(예:30초)마다 주기적으로 체크한다.

풀에서 연결을 얻기까지 대기시간을 줄인다(60초)

prepared-statement를 캐시하여 성능을 향상시킨다.

전자정부 프레임워크의 JBoss 데이터소스 사용 설정법

- 전자정부 프레임워크 기본 샘플은 DBCP를 사용하고 있음
- JBoss가 제공하는 데이터 소스를 사용하는 것이 성능, 관리 측면에서 유리함
- 아래와 같이 JBoss에서 설정한 데이터 소스 JNDI이름으로 변경하면 됨

- simple1/src/main/resources/egovframework/spring/com/context-datasource.xml

```
<!-- mysql
    <bean id="dataSource-mysql" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
        <property name="driverClassName" value="${Globals.DriverClassName}"/>
        <property name="url" value="${Globals.Url}" />
        <property name="username" value="${Globals.UserName}"/>
        <property name="password" value="${Globals.Password}"/>
    </bean>
-->
```

```
<jee:jndi-lookup id="dataSource-mysql" jndi-name="java:jboss/datasources/testDS" />
```

제품이나 서비스에 관한 문의

콜 센터 : 02-469-5426 (휴대폰 : 010-2243-3394)

전자메일 : sales@opennaru.com